parser generators other than YACC can be utilized and implemented by a system of the invention. By way of further example, the methods and systems according to the invention can be used to process HTML in contexts other than Internet-based, and client-server network applications. Thus, for example, they can be used to interpret UI objects displayed by a browser executing from an HTML file stored directly within a client device, e.g. a stand-alone HTML-based application. By way of further example, the invention can be used with mark-up languages other than HTML, e.g., XHTML, XUL.

In view of the foregoing, what is claimed is:

1. A method for identifying user interface (UI) objects in a markup-language stream, the method comprising the steps of:

A) scanning any of (i) the markup-language stream and (ii) a corresponding document object model (DOM) to generate tokens,

B) parsing the tokens based on a grammar to identify one or more UI objects.

2. The method of claim 1, wherein said markup-language stream drives a markup-language-based browser application, and wherein the scanning step includes scanning the DOM generated by a browser that displays that application.

3. The method of claim 1, wherein the scanning step includes identifying elements of the DOM by traversal thereof.

4. The method of claim 3, wherein the grammar is application-specific.

5. The method of claim 3, wherein the scanning step includes generating one or more tokens for each parsed DOM element.

6. The method of claim 3, wherein scanning step includes mapping DOM elements to tokens.

7. The method of claim 1, wherein the parsing step includes parsing the tokens according to the grammar to identify and distinguish among UI objects in the markup-language stream.

8. The method of claim 7, wherein said UI objects comprise user input fields, text fields, metatags, unprintable markup-language, and in-line images.

9. The method of claim 1, wherein the scanning and parsing steps are adapted to identify UI objects that correspond to elements displayed in the markup-language application.

10. The method of claim 9, wherein said parsing groups the tokens into syntactic structures that identify items displayed by the markup-language application.

11. The method of claim 9, wherein said step of scanning can include identifying similarly formatted markup-language elements based on their markup-language attributes such as classname, font size, style, tag color, and size.

12. The method of claim 9, wherein said objects comprise name, content, shape, location, and properties.

13. The method of claim 1, wherein said parser is built using automated tools such as YACC (yet another compiler-compiler).

14. The method of claim 13, wherein said parser is built by an automated parser generator tool that accepts a source input file containing a predefined grammar.

15. The method of claim 13, wherein said parser is built manually by hand-programming.

16. The method of claim 14, wherein the parsing is conducted by a LALR(1) parser.

17. The method of claim 14, wherein the parsing is conducted by a LR(1) parser.

18. The method of any of claims 1 - 17, wherein the markup language is any of HTML, XHTML and XUL.

19. A system for identifying user interface (UI) objects in markup-language-based applications comprising:

a scanner receiving markup-language DOM and generating one or more tokens for each DOM element, and

a parser coupled to the scanner receiving said tokens, and parsing said tokens based on a grammar, and generating a list of UI objects.

20. The system of claim 19, wherein the list of UI objects corresponds to elements displayed by the markup-language DOM.

21. The system of claim 20, wherein said UI objects comprise name, content, shape, location, and properties.

22. The system of claim 19, wherein the grammar is application-specific.

23. The system of claim 19, wherein said tokens are interpreted according to the grammar to identify and distinguish among UI objects of a markup-language application's display.

24. The system of claim 23, wherein the UI objects comprise user input fields, text fields, metatags, unprintable markup-language, and in-line images.

25. The system of any of claims 18 - 24, wherein the markup language is any of HTML, XHTML and XUL.